Name: Sahib Bajwa

ID: 107553096

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Exam 2 – S18 **Spring 2020, CU-Boulder**

Instructions: This quiz is open book and open note. You **may** post clarification questions to Piazza, with the understanding that you may not receive an answer in time and posting does count towards your time limit (30 min for 1x, 37.5 min for 1.5x, 45 min for 2x). Questions posted to Piazza **must be posted as PRIVATE QUESTIONS.** Other use of the internet, including searching for answers or posting to sites like Chegg, is strictly prohibited. Violations of these are grounds to receive a 0 on this quiz. Proofs should be written in **complete sentences. Show and justify all work to receive full credit.**

YOU MUST SIGN THE HONOR PLEDGE. Your quiz will otherwise not be graded. Honor Pledge: On my honor, I have not used any outside resources (other than my notes and book), nor have I given any help to anyone completing this assignment.

Your Name: Sahib Bajwa

Standard 18. Suppose we have n stairs to climb. You may choose to jump up either $1, 2$, or 3 stairs. Your goal is to count the number of ways to climb the stairs. Note that your starting position is on the ground floor and not on the first stair.

Is there a clear recursive structure in the problem that would be useful in designing an effective dynamic programming algorithm? That is, is dynamic programming a useful algorithmic technique for this problem? Clearly justify your answer.

I think that there is a way to design an effective dynamic programming algorithm to solve this problem. To solve this, I believe you first solve all possible ways to get up 3 stairs with our jumps of 1, 2, and 3 steps. We can then create our sub problems for our algorithm to solve. Our sub problems for this algorithm will be every possible set of 3 consecutive steps (0-3), (1-4), (2-5), and so on. We will then apply what we solved earlier and stored (all possible ways to get up 3 stairs with our jumps) to each sub-problem. We will then start to build our solution with compatible stair-jump sets (bottom-up) until we reach our solution.
Sub-problems: All sets of 3 consecutive steps.
Store solutions to sub-problems and apply to similar sub-problems.
Iterate on every possible combination and refer to stored solutions when we get to a similar sub-problem (which should be often).